The Dissertation Committee for Todd Wolman Geldon
Certifies that this is the approved version of the following dissertation:

COMPUTING THE TUTTE POLYNOMIAL OF HYPERPLANE ARRANGEMENTS

Committee:

Fernando Rodríguez-Villegas, Supervisor

Federico Ardila

John Tate

Jeffrey Vaaler

Felipe Voloch

COMPUTING THE TUTTE POLYNOMIAL OF HYPERPLANE ARRANGEMENTS

by

Todd Wolman Geldon, A.B.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2009

Acknowledgments

I would like to thank my advisor for always steering me in the right direction, my committee for all their help, and my friends and family for their support.

COMPUTING THE TUTTE POLYNOMIAL OF HYPERPLANE ARRANGEMENTS

Publication No. _____

Todd Wolman Geldon, Ph.D.
The University of Texas at Austin, 2009

Supervisor: Fernando Rodríguez-Villegas

We are studying the Tutte Polynomial of hyperplane arrangements. We discuss some previous work done to compute these polynomials. Then we explain our method to calculate the Tutte Polynomial of some arrangements more efficiently. We next discuss the details of the program used to do the calculation. We use this program and present the actual Tutte Polynomials calculated for the arrangements E_6, E_7, and E_8.

Table of Contents

List of Figures

Chapter 1

Introduction

The Tutte Polynomial has proven to be very useful in the world of graph theory. It is a two variable polynomial invariant that can be defined for graphs, can extend to the more general case of hyperplane arrangements, and can extend most generally to matroids. It is most commonly seen as a graph invariant, as isomorphic graphs have the same Tutte Polynomial. (However, the converse is not true.) Specific values of this polynomial encode useful data about the graph. For example, the Tutte Polynomial can express the number of forests, the number of spanning trees, and the number of connected spanning subgraphs of a graph. It also can specialize to the one variable chromatic polynomial. The difficulty with the Tutte Polynomial is that no efficient algorithm is known to compute it. For certain values, the computation can be done in polynomial-time. However, it has been shown that this is not true in general.

Less is known about the Tutte Polynomial of hyperplane arrangements. Federico Ardila developed some computation methods for the Tutte Polynomial in this situation using finite fields. We will expand on his methods by finding more efficient ways to calculate this polynomial for larger arrange-

ments. We demonstrate this method by showing a program in PARI/GP, and use it to calculate the Tutte Polynomial for the exceptional arrangements E_6, E_7, and E_8. Until very recently, these polynomials were not known.

We begin in Chapter 2 by discussing preliminaries about hyperplane arrangements, root systems, and Dynkin Diagrams. Chapter 3 contains the previous work by Ardila and the ideas behind our method. In Chapter 4, we discuss the algorithm we developed and implemented in GP to compute the Tutte Polynomial of certain arrangements. We give the actual Tutte Polynomials of the E_6, E_7, and E_8 arrangements. Finally, the appendix contains all the GP scripts used in our computation.

Chapter 2

Background

2.1 Hyperplane Arrangements

We begin by defining the terms we will be using.

Definition 2.1.1. Let $\mathbb{F}$ be a field and n a positive integer. An **affine hyperplane** in $\mathbb{F}^n$ is an affine subspace of $\mathbb{F}^n$ of codimension 1. A **hyperplane arrangement** A is a finite set of hyperplanes.

In general, a hyperplane can be described by a non-degenerate equation $a_1x_1+\ldots+a_nx_n = c$. We will usually refer to a hyperplane arrangement simply as an arrangement. Also, generally we will work over $\mathbb{F} = \mathbb{R}$.

Definition 2.1.2. If the hyperplanes in an arrangement have a non-empty intersection, then the arrangement is called **central**.

If an arrangement A is central, we can choose the coordinates such that each hyperplane contains the origin. For our purposes, we will assume that all central arrangements contain the origin.

Definition 2.1.3. The **rank** of a central hyperplane arrangement A is defined by $r(A) = n - \dim(\bigcap A)$. The rank function can be extended to all

arrangements by defining the rank of an arrangement to be the largest rank of a central subarrangement of that arrangement.

One well-known example of an arrangement is the braid arrangement, which we will later designate as having type A_n. The braid arrangement consists of the hyperplanes $x_i = x_j$ for all $i < j$. The braid space is the complement of these hyperplanes.

Definition 2.1.4. For an arrangement A, define the **intersection poset** $L(A)$ to be the set of all nonempty intersections of elements of A. The partial order on $L(A)$ is defined by reverse inclusion by $X \leq Y \Leftrightarrow Y \subseteq X$.

If A is the braid arrangement, then $L(A)$ is isomorphic to the lattice of partitions of n. (See [7] for a proof of this.) Here is the partition lattice for $n = 4$, and we can think of this as representing the intersection poset for the braid arrangement where a, b being in the same partition is equivalent to including the hyperplane $x_a = x_b$ in that intersection. (Image from [13].)

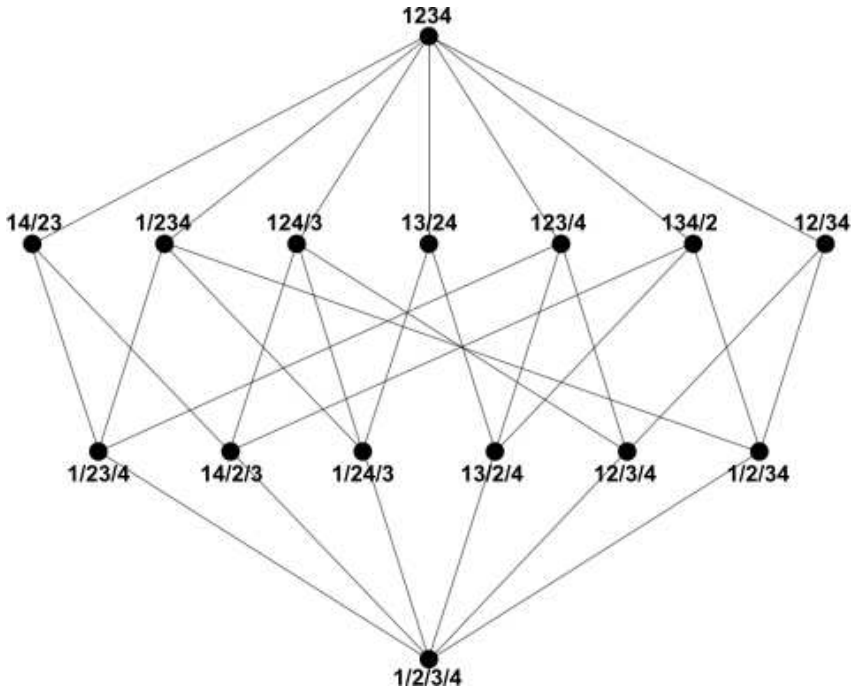

Figure 2.1: Partition lattice for n=4

2.2 Tutte Polynomial

The Tutte Polynomial is a two-variable polynomial that can be defined for graphs, hyperplane arrangements, and matroids. This can be done for each in several equivalent ways. One typical definition for graphs is as follows:

Definition 2.2.1. The Tutte Polynomial $T_G(q,t)$ of a graph G with edge set $E(G)$ is defined such that:

1. $E(G) = \{\} \implies T_G(q,t) = 1$
2. If G has a loop e, then $T_G(q,t) = tT_{G-e}(q,t)$.

3. If G has a bridge e, then $T_G(q,t) = qT_{G/e}(q,t)$.

4. For all other edges e, $T_G(q,t) = T_{G/e}(q,t) + T_{G-e}(q,t)$.

(Here G/e represents contracting e and $G - e$ represents deleting e.)

For a hyperplane arrangement, we can define the Tutte Polynomial $T_A(q,t)$ in a similar way using deletions and contractions. It is not clear that $T_G(q,t)$ or $T_A(q,t)$ are well-defined from this type of definition a priori. However, an equivalent and more convenient definition which is clearly well-defined is as follows:

Definition 2.2.2.

$$T_A(q,t) = \sum_{\substack{B \subseteq A \\ \text{B central}}} (q-1)^{r(A)-r(B)}(t-1)^{|B|-r(B)}$$

2.3 Finite Reflection Groups and Root Systems

For any hyperplane a defined over $\mathbb{F}^n$, let s_a be the reflection across that hyperplane. This reflection will fix a and it will send an orthogonal vector α to $-\alpha$. For any hyperplane arrangement, we will be examining the associated set of reflections and the group generated by these reflections. Specifically, we will look at the case when the group is finite (called a finite reflection group). We will think of these groups as groups generated by the corresponding hyperplane arrangements.

Given a vector space $\mathbb{F}^n$, a root system is a set S of non-zero vectors (called roots) and their negatives that satisfy the following properties:

1. S spans $\mathbb{F}^n$

2. $S \cap \mathbb{R}\alpha = \{\alpha, -\alpha\}$ for all $\alpha \in S$

3. $Ss_\alpha = S$ for all $\alpha \in S$

If we consider the standard Euclidean inner product on $\mathbb{F}^n$, then a root system is defined to be **crystallographic** if

$$2\frac{(\alpha_1, \alpha_2)}{\alpha_1, \alpha_1)} \in \mathbb{Z} \text{ for all } \alpha_1, \alpha_2 \in S.$$

This condition implies that a vector α_2 and its reflection $s_{\alpha_1}\alpha_2$ differ by an integral multiple of α_1, and we be only considering crystallographic root systems.

A root system is called **reducible** if it is the combination of two root systems which span mutually orthogonal subspaces of a common Euclidean space, and **irreducible** otherwise.

For a given hyperplane arrangement, we will consider the roots of the arrangement to vectors α that are orthogonal to some hyperplane in the arrangement. We will consider root systems that consist of these roots. These root systems have been well studied, and there are only a finite number of types of such systems which are irreducible.

A set of linearly independent roots that forms a basis for $\mathbb{F}^n$ with the property that every vector in S is a linear combination of elements of this set with all coefficients non-negative or all coefficients non-positive is called a set of simple roots for that system. It is not obvious a priori, but such a set of simple roots always exists ([5] 1.3).

2.4 Representations of Root Systems

There are four infinite families of root systems (A_n, B_n, C_n, D_n), and five exceptional cases $(E_6, E_7, E_8, F_4, G_2)$. We will describe one representation (as given in [5]) of each of the infinite families and the exceptional cases that we are interested in. In each case, let $\epsilon_1, \ldots, \epsilon_n$ represent the standard basis in $\mathbb{R}^n$.

$A_n, n \geq 1$: Let V be the subspace of $\mathbb{R}^{n+1}$ consisting of vectors whose coordinates sum to 0, and let the roots be those vectors that have integer coordinates and length $\sqrt{2}$. The roots will then be all $n(n+1)$ vectors of the form $\epsilon_i - \epsilon_j$, $j \neq i$. These correspond to the hyperplanes $\epsilon_i = \epsilon_j$. One set of simple roots is $\{\epsilon_i - \epsilon_{i+1}\}$ for $1 \leq i \leq n$.

$B_n, C_n, n \geq 2$: In $\mathbb{R}^n$, the roots of B_n are the vectors having integer coordinates and length either 1 or $\sqrt{2}$. This consists of the $2n$ vectors of the form $\pm\epsilon_i$ and the $2n(n-1)$ vectors of the form $\pm\epsilon_i \pm \epsilon_j$ for $i < j$. Thus the hyperplanes are of the form $\epsilon_i \pm \epsilon_j = 0$ and $\epsilon_i = 0$. One set of simple roots is $\epsilon_i - \epsilon_{i+1}$ for $1 \leq i \leq n-1$, and ϵ_n. C_n is isomorphic to B_n, with roots $\pm 2\epsilon_i$ instead of $\pm\epsilon_i$, and is its inverse root system.

$D_n, n \geq 4$: In $\mathbb{R}^n$, the roots are the vectors having integer coordinates and length $\sqrt{2}$. This consists of the $2n(n-1)$ vectors of the form $\pm\epsilon_i \pm \epsilon_j$ for $i < j$ that we saw in B_n. One set of simple roots is $\epsilon_i - \epsilon_{i+1}$ for $1 \leq i \leq n-1$, and $\epsilon_{n-1} + \epsilon_n$.

E_8: In $\mathbb{R}^8$, we consider the following 240 roots: there are 112 roots of the form $\pm\epsilon_i \pm \epsilon_j$ for $i < j$, and there are 128 roots of the form $\frac{1}{2}\sum_{i=1}^{8} \pm\epsilon_i$

such that there are an even number of plus signs. We note that all roots have constant length $\sqrt{2}$. One set of simple roots is: $\epsilon_i - \epsilon_{i-1}$ for $2 \leq i \leq 7$, $\epsilon_1 + \epsilon_2$, and $\frac{1}{2}(\epsilon_1 + \epsilon_8 - \sum_{i=2}^{7} \epsilon_i)$.

E_7: E_7 is easily defined as a subset of E_8. It contains the following 126 roots in $\mathbb{R}^8$: there are 60 roots of the form $\pm\epsilon_i \pm \epsilon_j$ for $1 \leq i < j \leq 6$, there are 2 roots of the form $\pm\epsilon_7 - \epsilon_8$, and there are 64 roots of the form $\pm\frac{1}{2}(\epsilon_7 - \epsilon_8 + \sum_{i=1}^{6} \pm\epsilon_i$ such that the sum contains an odd number of minus signs. Then one set of simple roots is the same as for E_8, removing $\epsilon_7 - \epsilon_6$.

E_6: We define E_6 as a subset of E_8 as well. It contains the following 72 roots over $\mathbb{R}^8$: there are 40 roots of the form $\pm\epsilon_i \pm \epsilon_j$ for $1 \leq i < j \leq 5$, and there are 32 roots of the form $\pm\frac{1}{2}(\epsilon_8 - \epsilon_7 - \epsilon_6 + \sum_{i=1}^{5} \pm\epsilon_i$ such that the sum contains an odd number of minus signs. The one set of simple roots is the same as for E_8, removing $\epsilon_7 - \epsilon_6$ and $\epsilon_6 - \epsilon_5$.

One advantage of defining E_7 and E_6 over $\mathbb{R}^8$ is that we can see the copy of them in E_8. We can also define E_7 and $\mathbb{R}^7$ and E_6 in $\mathbb{R}^6$ using a change of basis, by using the simple roots as our basis. We will use this change of basis later when we are doing our calculations for E_7 and E_6.

2.5 Dynkin Diagrams

Dynkin diagrams are a way to represent a system based on the relationship of a set of simple roots of that system. This diagram is independent of the choice of simple roots. It is a graph in which the vertices of the diagram correspond to the simple roots. An edge is drawn between two vertices if and only if the

corresponding roots are not orthogonal. To designate the angle between non-orthogonal roots, different edges are used. A single edge represents an angle of 120 degrees, a double edge represents an angle of 135 degrees, and a triple edge represents an angle of 150 degrees.

Another interpretation of Dynkin diagrams relates to the actions of the reflection group. This group is generated by the simple roots, and we can look at the order of products of simple roots. Specifically, for any two simple roots α and β, define $m(\alpha, \beta)$ to be the order of the associated hyperplane reflections s_α and s_β, so $(s_\alpha s_\beta)^{m(\alpha,\beta)} = 1$. Then an edge in the Dynkin diagram is drawn between two vertices exactly when the associated order $m \geq 3$. Specifically, a single edge represents order 3, a double edge represents order 4, and a triple edge represents order 5.

The Dynkin Diagrams of the root systems are well known, and are as follows (image from [14]):

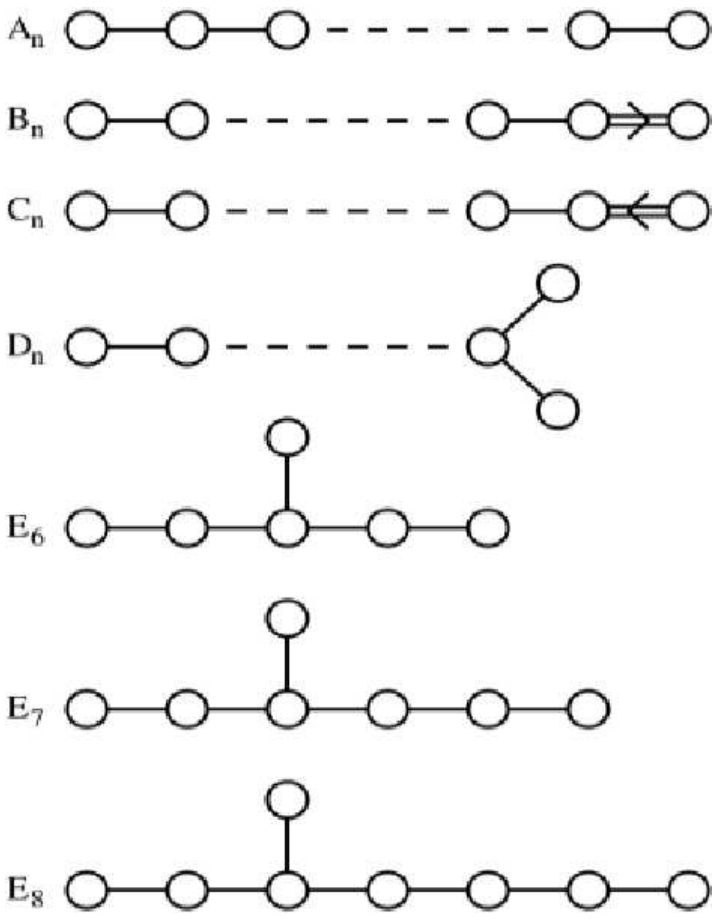

Figure 2.2: Dynkin Diagrams of Root Systems

Chapter 3

Computing the Tutte Polynomial

3.1 Previous Methods

Recently, de Concini and Procesi created a method to compute the Tutte Polynomial that involves what are known as the internal and external activities of a basis. Suppose we have a list of vectors X in $\mathbb{R}^n$ which span $\mathbb{R}^n$. Fix a total ordering on X. Consider a basis $B \subset X$, where B inherits the total order $B = \{x_{i_1}, \ldots, x_{i_n}\}$.

Definition 3.1.1. An element $a \in X$ is externally active with respect to B if for some $e \leq n$, we have $a < x_{i_e}$ and the list $(a, x_{i_e}, \ldots, x_{i_n})$ is linearly dependent. An element $b \in B$ is internally active with respect to B if there is no element a in X preceding b such that $\{a\} \cup (B/\{b\})$ is a basis for X. Then $e(B)$ and $i(B)$ are the number of externally and internally active elements with respect to B, and are known as the **external activity** and **internal activity** of B.

Tutte proved that

$$T(q,t) = \sum_{B \text{ basis}} q^{i(B)} t^{e(B)}$$

In [3], de Concini and Procesi use this idea to determine ways of calculating the Tutte Polynomial. Using this formula directly is very computationally intensive, but they were able to use this idea and other results to find more efficient ways of performing the calculation.

3.2 Ardila's Finite Field Method

3.2.1 Coboundary Polynomial

In [1], Ardila introduced a method of computing the Tutte Polynomial for hyperplane arrangements. His results use a transformation of the Tutte Polynomial into the coboundary polynomial $\bar{\chi}_A(q,t)$, defined as follows:

Definition 3.2.1.

$$\bar{\chi}_A(q,t) = \sum_{\substack{B \subseteq A \\ \text{B central}}} q^{r(A)-r(B)}(t-1)^{|B|}$$

There is a simple transformation between the polynomials:

$$\bar{\chi}_A(q,t) = (t-1)^r T_A\left(\frac{q+t-1}{t-1}, t\right)$$

and

$$T_A(q,t) = \frac{1}{(t-1)^r}\bar{\chi}_A((q-1)(t-1), t)$$

Thus being able to compute $\bar{\chi}_A(q,t)$ is essentially equivalent to being able to compute $T_A(q,t)$.

3.2.2 Reductions

We will define an **integral arrangement** to be an arrangements of hyperplanes whose defining equations have integer coefficients. Ardila created a method to compute Tutte polynomials of integral arrangements by looking at the reductions of the arrangements over finite fields. Specifically, let A be an integral arrangement defined over $\mathbb{Z}^n$, and let $q = p^k$ be a prime or a prime power. Then A can be reduced to an arrangement A_q over $\mathbb{F}_q^n$ by regarding the defining equations as equations over $\mathbb{F}_q$. We first consider what conditions we require for this reduction to be useful.

If A and A_q are combinatorially isomorphic arrangements, then we will say that we have a **valid** reduction, and we will call p a **valid** prime. However, for certain values of q we might not get a valid reduction. Some clear situations where the reduction is not valid are when two hyperplanes reduce to the same hyperplane or a hyperplane becomes degenerate. There are other, more subtle possibilities as well. For the reduction to be valid, the following property must hold: Given an integral arrangement A and prime power $q = p^k$, for any subset X of the equations of A, the span of X in $\mathbb{F}_q$ must have the same dimension as the span of X in $\mathbb{R}$.

For X to have the same dimension in $\mathbb{F}_q$, the determinants of all minors of the matrix of coefficients of the hyperplanes must be non-zero mod p. Thus certainly if we choose p to be a prime larger than all these determinants, we can guarantee that we have a valid reduction. In fact, we will see details later of exactly which primes are valid for the E_6, E_7, and E_8 arrangements.

3.2.3 Reduction Theorem

Assuming we have a valid reduction, Ardila proved the following theorem.

Theorem 3.2.1 ([1]). *Let A be a $\mathbb{Z}$-arrangement in $\mathbb{R}^n$. Let q be a power of a large enough prime, and let A_q be the induced arrangement in $\mathbb{F}_q^n$. Then*

$$q^{n-r}\bar{\chi}_A(q,t) = \sum_{x \in \mathbb{F}_q^n} t^{h_q(x)}$$

where $h_q(x)$ denotes the number of hyperplanes of A_q that x lies on.

Ardila used this theorem to compute the coboundary polynomials of the Coxeter arrangements of type A_n, B_n, and D_n. In each case, he did this by computing the exponential generating function for the polynomials. However, this technique appears to only work for these specific types and is not a general method for all arrangements.

3.3 Computing Tutte Polynomials for Larger Arrangements

In principle, one could use Ardila's Theorem to compute the coboundary polynomial for any arrangement with integral coefficients (more generally, with rational coefficients). By choosing a valid prime power q_0, one could compute the right hand side directly by summing over all points in $\mathbb{F}_{q_0}^n$. This would compute the polynomial $\bar{\chi}_A(q_0, t)$. Computing this polynomial for enough values of q_0 would allow us to use Lagrange interpolation to find $\bar{\chi}_A(q,t)$, thus giving the full coboundary polynomial of the arrangement.

However, actually computing the Tutte Polynomial of certain arrangements this way is too computationally intensive to be practical. For example, E_8 has 120 hyperplanes, and this method requires checking whether each of the q^8 points is on each of the 120 hyperplanes. In order to interpolate completely, we will show later that we would need to do this computation for at least some $q \geq 31$. We propose a method for computing the coboundary polynomial that is not as computationally intensive.

3.3.1 Computation Method

The idea of the method is as follows. For integers p and q (not necessarily prime), and given an arrangement, suppose we want to count the number of points on each of the hyperplanes if we reduce the arrangement mod pq. This is equivalent to counting the points mod p and looking at the cosets mod q. We will show that this works not only for finite fields, but finite rings as well, which is why we do not require p and q to be prime.

The main result we present is the following: Suppose p and q are valid primes or products of valid primes for an arrangement A, and let t be a variable. Then

$$(pq)^{n-r(A)}\bar{\chi}_A(pq,t) = \sum_{x\in(\mathbb{Z}/p\mathbb{Z})^n} q^{n-r(A_x)}\bar{\chi}_{A_x}(q,t)$$

where A_x represents the subset of hyperplanes of A that x lies on mod p.

We will use this result to actually compute coboundary polynomials for the E_6, E_7, and E_8 arrangements. As we will show using Dynkin diagrams, these arrangements have a very limited number of subarrangements. Thus

if A is one of these arrangements, then there are a very limited number of possibilities for the A_x subset.

We can thus pick a small valid value for q and calculate $\bar{\chi}_{A_x}(q,t)$ for all possible subsets A_x. Then for different values of p, we can compute $\bar{\chi}_A(pq,t)$ using this formula, and this is not computationally intensive as long as p is relatively small. Essentially, this gives us many more options for calculating specific instances of the coboundary polynomial. We then do this computation for enough values of pq so that we can interpolate to get the general polynomial.

3.3.2 Proof of Method

We will start by proving the following lemma:

Lemma 3.3.1. *Suppose we have $A \in \mathbb{Z}^{m\times n}$ of rank n, and let p and q be relatively prime to the determinants of all minors of A. Let $x \in \mathbb{Z}^n$ such that $Ax \equiv 0 \bmod p$, and let $b \in \mathbb{Z}^n$ be such that $Ax = pb$. Then there exists some $r \in \mathbb{Z}^n$ such that $b + Ar \equiv 0 \bmod q$.*

Proof. First, if $x \equiv 0 \bmod p$ then $r = 0$ is clearly a solution. For any $x \in \mathbb{Z}^n, x \not\equiv 0 \bmod p$, consider A. Let S be any subset of n rows of A. If S has nonzero determinant, then let $M = S^{-1}$, so we can write $M = \frac{1}{\det(S)}M'$ where M' has integral entries. We know $Ax = pb$, so let x' and b' represent the appropriate subsets of x and b such that $Sx' = pb'$. Then $x' = Mpb' = \frac{1}{\det S}M'pb'$. However, we know that $\gcd(p, \det S) = 1$, so $p|Mpb'$. However, because $x \not\equiv 0 \bmod p$, p cannot divide x'. Thus we have a contradiction, so we know $\det(S) = 0$.

This is true for any n rows of A, so A has fewer than n independent rows. Because of this, we see that there must be a solution r to this system of equations.

□

We note that the restrictive conditions on the matrix A are essential, because the lemma is not true in general. An example of when a minor is 0 and r does not exist, found experimentally, is as follows: Let $A = E_8$, $p = q = 5$, $x = [1, 1, 1, 1, 1, 1, 1, 1]$. Then x is on 20 hyperplanes of A, but for any z, the maximal number of hyperplanes containing $x + pz$ is 16, and thus our desired r does not exist. However, p and q are not relatively prime to all minors of A as required by the conditions in the lemma.

For any n, let $h_n(y)$ represent the number of hyperplanes of A that y is on mod n. We now prove the following lemma:

Lemma 3.3.2. *Let A be a central arrangement in $\mathbb{R}^n$, let p and q be products of valid odd primes for A with $\gcd(p, q) = 1$, and let t be a variable. Then*

$$\sum_{y \in (\mathbb{Z}/pq\mathbb{Z})^n} t^{h_{pq}(y)} = \sum_{x \in (\mathbb{Z}/p\mathbb{Z})^n} q^{n-r(A_x)} \bar{\chi}_{A_x}(q, t)$$

where A_x represents the subset of hyperplanes of A that x lies on mod p.

Proof. Consider all points $y \in (\mathbb{Z}/pq\mathbb{Z})^n$. We can write $(\mathbb{Z}/p\mathbb{Z})^n \times (\mathbb{Z}/q\mathbb{Z})^n \simeq (\mathbb{Z}/pq\mathbb{Z})^n$ by $(x \bmod p, z \bmod q) \mapsto x + pz \bmod pq$. Thus we can think of $y = x + pz$ where $x \in (\mathbb{Z}/p\mathbb{Z})^n$ and $z \in (\mathbb{Z}/q\mathbb{Z})^n$.

For each such y, we wish to determine $h_{pq}(y)$, the number of hyperplanes that y lies on mod pq. Fix some $x \in (\mathbb{Z}/p\mathbb{Z})^n$, and this determines some subset of hyperplanes A_x. We want to look at the points $y = x + pz$ as described above, with fixed x but varying z, and determine for each one how many planes of A it is on mod pq. It is clear that if y is on some hyperplane of A mod pq, then certainly x must be on that hyperplane mod p, and thus that hyperplane must be in A_x. Thus it is enough to only consider the subset A_x.

We know $A_x x = 0 \bmod p$, so let $A_x x = pb$. Then

$$A_x(x + pz) = A_x x + A_x pz = p(b + A_x z)$$

Let b_i and z_i denote the ith coordinates of b and z respectively, and let A_{xi} denote the ith row of A_x. Then $h_{pq}(y) = \#\{i : p(b_i + A_{xi} z_i) = 0 \bmod q\}$.

We would like to find a value $z = r$ such that $y = x + pr$ is on all the hyperplanes of A_x. Thus we want $h_{pq}(x + pr) = \#A_x$ so $b + A_x r = 0 \bmod q$. We know that A_x is a subarrangement of A such that p and q are relatively prime to all of its minors, so by lemma 3.3.1, such an r must exist.

Given that r exists, we can now write $\{z : z \in (\mathbb{Z}/q\mathbb{Z})^n\} = \{z + r : z \in (\mathbb{Z}/q\mathbb{Z})^n\}$, and thus

$$\begin{aligned}
\sum_{z \in (\mathbb{Z}/q\mathbb{Z})^n} t^{h_{pq}(x+pz)} &= \sum_{z \in (\mathbb{Z}/q\mathbb{Z})^n} t^{h_{pq}(x+p(z+r))} && (3.1)\\
&= \sum_{z \in (\mathbb{Z}/q\mathbb{Z})^n} t^{h_{pq}(pz)} && (3.2)\\
&= \sum_{z \in (\mathbb{Z}/q\mathbb{Z})^n} t^{h_q(z)} && (3.3)
\end{aligned}$$

so

$$\sum_{y\in(\mathbb{Z}/pq\mathbb{Z})^n} t^{h_{pq}(y)} = \sum_{x\in(\mathbb{Z}/p\mathbb{Z})^n}\sum_{z\in(\mathbb{Z}/q\mathbb{Z})^n} t^{h_{pq}(x+pz)} \tag{3.4}$$

$$= \sum_{x\in(\mathbb{Z}/p\mathbb{Z})^n}\sum_{z\in(\mathbb{Z}/q\mathbb{Z})^n} t^{h_q(z)} \tag{3.5}$$

where the function h_q is based on the value of x chosen, because x determines the hyperplanes in A_x.

Thus by Ardila's Theorem, we see that

$$\sum_{y\in(\mathbb{Z}/pq\mathbb{Z})^n} t^{h_{pq}(y)} = \sum_{x\in(\mathbb{Z}/p\mathbb{Z})^n} q^{n-r(A_x)}\bar{\chi}_{A_x}(q,t)$$

which completes the proof.

□

We prove one more lemma about the intersections of hyperplanes, now in the context of more general rings of the form $R = (\mathbb{Z}/s\mathbb{Z})$, where s is the product of valid odd primes.

Lemma 3.3.3. *Let A be an integral arrangement reduced over the ring $R = (\mathbb{Z}/s\mathbb{Z})$, where s is the product of valid odd primes for A. Let $H_1, \ldots, H_k$ be the hyperplanes in A. For any $I \subseteq \{1, \ldots, k\}$, let $H_I = \bigcap_{i\in I} H_i$. Then $H_I \simeq \mathbb{A}^{d_I}$ for some $d_I \geq 0$, where $\mathbb{A}$ represents an affine space.*

Proof. We prove this by induction on $|I|$.

Suppose $|I| = 1$, so $I = \{i\}$. Then H_i is the set of solutions to $l \cdot x = 0$ where $l = (l_1, \ldots, l_n) \in \mathbb{Z}^n$. Thus for any ring R, we can write the points on the hyperplane $H_i(R) = \{(x_1, \ldots, x_n) | x_j \in R, \sum_{j=1}^{n} l_j x_j = 0\}$.

Let $R = (\mathbb{Z}/s\mathbb{Z})$. We assume s is the product of valid odd primes for A, so some l_j must be a unit in R. WLOG assume that l_n is a unit. Thus we have a bijection between $H_i(R) \leftrightarrow R^{n-1}$ given by $(y_1, \ldots, y_{n-1}, x_n) \leftrightarrow (y_1, \ldots, y_{n-1})$ where $x_n = -l_n^{-1} \sum_{j=1}^{n-1} l_j y_j$. We can think of $H_i(R)$ as being an affine space of dimension $n - 1$.

Now for a set I, consider some H_i with $i \in I$. By hypothesis, $H_{I/i}(R) \simeq \mathbb{A}^{d'}$ for some $d' \geq 0$, and $H_i \simeq \mathbb{A}^{n-1}$. Thus $H_I(R) = H_i(R) \cap H_{I/i}(R) = \mathbb{A} \cap \mathbb{A}^{d'}$. Depending on the independence of $H_i(R)$ in $H_{I/i}(R)$, this intersection of affine spaces will either be $\mathbb{A}^{d'}$ or $\mathbb{A}^{d'-1}$. Thus we see that $H_I \simeq \mathbb{A}^d$ for some d.

□

Now these lemmas let us prove the following result:

Proposition 3.3.4. *Let s be the product of valid odd primes for A, and let t be a variable. Then there exists a polynomial $C(x, y) \in \mathbb{Z}[X, Y]$ such that*

$$C(s, t) = \sum_{y \in (\mathbb{Z}/s\mathbb{Z})^n} t^{h_s(y)}$$

Proof. Let A be an integral arrangement in $\mathbb{R}^n$. We will prove this statement by induction on the dimension n.

Suppose $n = 1$, so all hyperplanes are simply points. Suppose there are k distinct hyperplanes, and let n_i represent the number of hyperplanes that

occur i times. Then $C(s,t) = \sum_{y \in (\mathbb{Z}/s\mathbb{Z})^n} t^{h_s(y)} = (s-k) + \sum_i n_i t^i$ is indeed a polynomial.

From Katz in [6], we use the following: Given a noetherian ring R, we denote by (Sch/R) the category of separated R-schemes of finite type, morphisms being the R-morphisms. We denote by K_0(Sch/R) its Grothendieck group. By definition, K_0(Sch/R) is the quotient of the free abelian group on elements [X], one for each separated R-scheme of finite type, by the subgroup generated by all the relation elements

$$[X] - [Y] \text{ ,whenever } X^{red} \equiv Y^{red},$$

and

$$[X] - [X \backslash Z] - [Z] \text{ ,whenever } Z \subset X \text{ is a closed subscheme.}$$

Then if X is a finite union of locally closed subschemes Z_i, then in K_0(Sch/R) we have the inclusion-exclusion relation

$$[X] = \sum_i [Z_i] - \sum_{i<j} [Z_i \cap Z_j] + \cdots$$

We use this as follows. Suppose $n > 1$. Consider the space $X = \cup_i H_i$, i.e. the points that are on at least one hyperplane. By the ideas above, we can write:

$$X = \sum_i H_i - \sum_{i,j} (H_i \cap H_j) + \cdots \pm \cap_i H_i$$

in $K_0(\mathbb{Z})$.

Now, passing to our ring R via the map $\mathbb{Z} \to R$, using Lemma 3.3.3, and taking cardinalities yields the following:

$$|X(R)| = \sum_i |\mathbb{A}^{d_i}| - \sum_{i,j} |\mathbb{A}^{d_{(i,j)}}| + \cdots \pm |\mathbb{A}^{d_{(1,2,\ldots,k)}}|$$

We know that $|\mathbb{A}^d| = s^d$. Thus we can write $C_1(s) = |X(R)|$ for some polynomial $C_1(x) \in \mathbb{Z}[X]$. Since there are s^n points overall, this gives us $C(s,0) = s^n - C_1(s)$, which is a polynomial as required.

Now consider some hyperplane H_i, and let X_i be the arrangement created by intersecting A with H_i. By restricting to H_i, we create an arrangement of dimension $n-1$. Thus by induction, we know $C_{X_i}(s,t) = \sum_{y \in (\mathbb{Z}/s\mathbb{Z})^n} t^{h_s(y)}$ is a polynomial, where h_s here refers to the arrangement X_i.

We do this for all hyperplanes and get a polynomial C_{X_i} for each H_i. If we add these polynomials, then every point x is counted in the sum with multiplicity based on the number of hyperplanes $h_s(x)$ that contain it. Each time, it is counted in the polynomial as a term $t^{h_s(x)}$, which is the same regardless of which H_i we used to calculate the polynomial.

Thus we can again we use the principle of inclusion/exclusion based on the poset of arrangements to correctly count all points with multiplicity one. Specifically, for $I \subset \{1, \ldots, k\}$, again let $H_I = \cap_{i \in I} H_i$, let and C_I be the polynomial as described above from intersecting $A \backslash H_I$ with H_I. This C_I can be thought of as successively restricting A to each hyperplane in I, and each time we get a polynomial by induction. Then by inclusion/exclusion,

$$C(s,t) = C(s,0) + \sum_i C_{X_i} - \sum_{i,j} C_{\{i,j\}} + \sum_{i,j,k} C_{\{i,j,k\}} - \ldots \pm C_I$$

.

Clearly, $C(s,t)$ is the result of the addition and subtraction of various polynomials and so the result is also a polynomial.

□

This theorem shows that the sum $\sum_{y \in (\mathbb{Z}/s\mathbb{Z})^n} t^{h_s(y)}$ is a polynomial in the finite ring setting as well as the finite field setting. Thus we can now show our main result, which essentially states that this polynomial $C(x,y)$ must be the coboundary polynomial in this setting, generalizing Ardila's Theorem.

Corollary 3.3.5. *Let p and q be valid primes or products of valid primes for A, and let t be a variable. Then*

$$(pq)^{n-r(A)} \bar{\chi}_A(pq,t) = \sum_{x \in (\mathbb{Z}/p\mathbb{Z})^n} q^{n-r(A_x)} \bar{\chi}_{A_x}(q,t)$$

Proof. By Ardila's Theorem, we know that for all valid primes p,

$$p^{n-r} \bar{\chi}_A(p,t) = \sum_{x \in \mathbb{F}_p^n} t^{h_p(x)}$$

By Proposition 3.3.4, we know that $\sum_{y \in (\mathbb{Z}/s\mathbb{Z})^n} t^{h_s(y)}$ must be a polynomial in s for all valid s.

Thus we see that the right hand side of Ardila's Theorem, $\sum_{x \in \mathbb{F}_p^n} t^{h_p(x)}$, is always a polynomial. We know its value for an infinite number of primes, and so it must be true that

$$s^{n-r} \bar{\chi}_A(s,t) = \sum_{x \in (\mathbb{Z}/s\mathbb{Z})^n} t^{h_s(x)}$$

for any s.

Thus by lemma 3.3.2, we let $s = pq$ and see that

$$(pq)^{n-r(A)} \bar{\chi}_A(pq,t) = \sum_{x \in (\mathbb{Z}/p\mathbb{Z})^n} q^{n-r(A_x)} \bar{\chi}_{A_x}(q,t)$$

which we were trying to show.

□

Chapter 4

Algorithm

4.1 Overview

There are several phases to the actual computation of the coboundary polynomial. The first step is determining which primes are valid for use and will represent valid reductions. For a sufficient number of valid prime powers q, we can compute the coboundary polynomial in $\mathbb{F}_q$ using Ardila's Theorem. Then we can use our modification of this theorem to compute this polynomial in $(\mathbb{Z}/q\mathbb{Z})$ for any q that is the product of valid primes. Finally, we can interpolate the polynomials we get to find the general coboundary polynomial.

4.2 Valid Prime Powers

To determine which primes are valid, we must look at the determinants of the arrangement and of all subarrangements. To see what the possible subarrangements are, we will use the following lemma:

Lemma 4.2.1. *Every subarrangement of E_6, E_7, and E_8 can be represented by a subdiagram of the Dynkin Diagram of the original arrangement.*

Proof. Consider the initial arrangement and take a set of simple roots. The

relationship of the simple roots is invariant, and this determines the Dynkin Diagram. For any subarrangement, we can represent it by a subset of the simple roots, which will have the same relationship with each other. Thus we can represent this by a subdiagram of the original Dynkin Diagram. □

Thus to look at the subarrangements we must simply look at all subdiagrams of the Dynkin Diagram. Consider the arrangement E_6. The only possible subdiagrams (and therefore subarrangements) are A_n $(1 \leq n \leq 5)$ and D_n $(n = 4, 5)$. It is well known that the determinant of A_n is $n + 1$, the determinant of D_n is 4, and the determinant of E_6 itself is 3. Thus the only possible invalid primes are $p = 2, 3, 5$. However, A_4 is primitive in E_6, and this is the only subarrangement that that could cause 5 to be invalid, and so $p = 5$ is in fact a valid prime.

We can use the same logic for E_7 and E_8. The determinants of E_7 and E_8 are 2 and 1, respectively. For E_7, we must include E_6, A_6, and D_6 as possible subarrangements, as well as all subarrangements of E_6. Similarly for E_8 we must include E_7, A_7, and D_7, as well as all subarrangements of E_7. Thus we can see that the only possible invalid primes for both E_7 and E_8 are $p = 2, 3, 5, 7$, where the addition of 7 comes from the subarrangement A_6. Similarly, it turns out that A_4 is primitive in E_7, and A_6 is primitive in E_8. Thus $p = 5$ is valid for E_7, and $p = 7$ is valid for E_8.

Thus we see that all primes $p \geq 5$ and powers of these primes must be valid for E_6 and E_7, and all primes $p \geq 7$ and their powers must be valid for E_8.

4.3 Preliminary Computations

Our goal is now to calculate the coboundary polynomial for an arrangement A over the ring with $q = q_1 q_2$ elements, where both q_1 and q_2 are products of valid primes. First we must do some precomputations that we will use in the actual computation.

4.3.1 Calculation for Primes

For each of E_6, E_7, and E_8, we first have to determine the set of hyperplanes corresponding to the root system. We do this by defining each arrangement over $\mathbb{R}^8$, and then performing a change of basis to define each E_n over $\mathbb{R}^n$ as defined earlier. This is done in createEPlanes.

The next step is to use Ardila's Theorem to compute the coboundary polynomial of our arrangement for several small valid primes p. This is a straightforward application of his formula, and the program can be found in Primes.gp.

4.3.2 Subarrangements

Given a choice of q_2, we will next compute the coboundary polynomials of all subarrangements based on this choice. We first use [[1], Thm 4.1 and 4.3] to compute the coboundary polynomial of A_n and D_n for small values of n.

Next, the routine CreateSubPolys will combine the data about all possible subarrangements. In this routine, all possible combinations of subarrangements are considered, based on the Dynkin Diagram of our arrangement.

We use the precomputed polynomials for arrangements of type A, D, and E, because every subarrangement is a sub-diagram of the given diagram, and then consider all possible direct sums of these arrangements that give a subdiagram of our arrangement. In each case, we store the polynomial, the number of planes involved, and the number of sets of three dependent planes. Note that the subarrangement based on when x is the origin is the entire arrangement, and thus must appear in our list. This is why we needed to calculate the coboundary polynomial of this arrangement for q_1.

4.4 Reductions

Once the preliminary work has been determined, we can now compute the coboundary polynomial for our arrangement for the given value of q using Corollary 3.3.5. This is done in GetCoPoly, and works as follows:

It calculates the coboundary polynomial for our arrangement over the ring of size $q = q_1q_2$, where we assume that we already know the coboundary polynomial for q_1. We also assume that we have the necessary precomputed values from above. To use Ardila's Theorem, what we would normally want to do is run through all points, and for each point compute the number of hyperplanes that contain that point. However, we know by Corollary 3.3.5 that we can instead look at all points in $(\mathbb{Z}/q_1\mathbb{Z})$, and for each one we will consider all the points in $(\mathbb{Z}/q_1q_2\mathbb{Z})$ that are congruent to our point mod q_2. These points must form a subarrangement of our original arrangement, and thus the count we want must give us the coboundary polynomial for that subarrangement. Because we have already determined all possible subarrangements and

precomputed the coboundary polynomial for each of them, we simply must find the applicable one for each point of $(\mathbb{Z}/q_1\mathbb{Z})$.

Now we have reduced the problem to determining which subarrangement we have for each of our points in $(\mathbb{Z}/q_1\mathbb{Z})$. For each such point x, we look at the planes that x goes through mod q_1. Now, the number of planes must be consistent with the number of planes in the subarrangement, and this is often sufficient for determining what subarrangement we have. If there are multiple subarrangements with that same number of planes, then, we use the number of sets of three planes that are dependent. These two pieces of data turn out to completely characterize which subarrangement we have for E_6, E_7, and E_8. This gives us the coboundary polynomial for each point x, and we use this to find the coboundary polynomial of the entire arrangement.

4.5 Interpolation

Once we have found polynomials specified at enough values of q, we can interpolate to find the general coboundary polynomial. The number of polynomials we need is simply one more than the largest possible degree of q. In the definition of the coboundary polynomial, the exponent of q is $r(A) - r(B)$, which has a maximal value of $r(A)$, the rank of the original arrangement. Thus for the arrangement E_i we only need $i + 1$ polynomials.

The interpolation is done in the file interp.gp. For each k, it looks at the coefficient of t^k of each of the $i + 1$ polynomials to determine the general coefficient of t^k. We did this calculation with additional values of q as a check

for consistency.

4.6 Values Used

To compute the coboundary polynomial for E_6, we were able to purely use Ardila's Theorem because the size of the arrangement is sufficiently small. We computed the specializations for primes up to $p = 37$, and interpolated to find the coboundary polynomial. We also did the computation for all products of primes $p = 5$, $p = 7$, and $p = 11$ to demonstrate consistency.

To compute the coboundary polynomial for E_7, we needed to use our reduction method. We computed the coboundary polynomial for primes up to $p = 19$, and then used the reduction method on all products of primes $p = 7, 11, 13, 17, 19$.

To compute the coboundary polynomial for E_8, we needed to use our reduction method. The running time of the algorithm on E_8 is $\mathcal{O}(q_2^8)$, which is large enough that we wanted to keep q_2 minimal. We computed the coboundary polynomial for primes up to $p = 13$, and then used the reduction method on $q = 7^m$, $q = 11 \cdot 7^m$, and $q = 13 \cdot 7^m$ for several small values of m so that we could use $q_2 = 7$ in all cases.

4.7 Results

4.7.1 Coboundary Polynomials

Using the program, we found the following coboundary polynomials:

$\bar{\chi}_{E_6}(q,t) = \sum_{i=0}^{36} C_i t^i$ where the C_i's are the following functions of q

($C_i = 0$ if not listed):

$$
\begin{aligned}
C_{36} &= 1 \\
C_{20} &= 3^3(q-1) \\
C_{15} &= 2^2 3^2 (q-1) \\
C_{12} &= 3^2 5^1 (q-1)(q-2) \\
C_{11} &= 2^3 3^3 (q-1) \\
C_{10} &= 2^3 3^3 (q-1)(q-3) \\
C_7 &= 2^2 3^2 5^1 (q-1)(3q-7) \\
C_6 &= 2^1 3^1 5^1 (q-1)(9q^2 - 59q + 88) \\
C_5 &= 2^2 3^3 5^1 (q-1)(q-4) \\
C_4 &= 2^4 3^2 5^1 (q-1)(q-4)(q-5) \\
C_3 &= 2^2 3^1 5^1 (q-1)(q-4)(q-5)(2q-1) \\
C_2 &= 2^1 3^3 5^1 (q-1)(q-4)(q-5)(q-7) \\
C_1 &= 2^2 3^2 (q-1)(q-4)(q-5)(q-7)(q-8) \\
C_0 &= (q-1)(q-4)(q-5)(q-7)(q-8)(q-11)
\end{aligned}
$$

$\bar{\chi}_{E_7}(q,t) = \sum_{i=0}^{63} C_i t^i$ where the C_i's are the following functions of q ($C_i = 0$ if not listed):

$$
\begin{aligned}
C_{63} &= 1 \\
C_{36} &= 2^2 7^1 (q-1) \\
C_{30} &= 3^2 7^1 (q-1) \\
C_{21} &= 2^1 3^2 37^1 (q-1) \\
C_{20} &= 2^1 3^3 7^1 (q-1)(q-3) \\
C_{16} &= 2^4 3^2 7^1 (q-1) \\
C_{15} &= 2^5 3^1 7^1 (q-1)(2q-7) \\
C_{13} &= 3^2 7^1 (q-1)(15q-13) \\
C_{12} &= 3^2 5^1 7^1 (q-1)(q-3)(q-5) \\
C_{11} &= 2^5 3^3 7^1 (q-1)(q-4) \\
C_{10} &= 2^4 3^2 7^1 (q-1)(2q^2-18q+45) \\
C_9 &= 2^4 3^2 5^1 7^1 (q-1)(q-5) \\
C_8 &= 2^3 3^3 5^1 7^1 (q-1)(q-5) \\
C_7 &= 2^2 3^2 5^1 7^1 (q-1)(q-5)(7q-29) \\
C_6 &= 2^2 3^1 5^1 7^1 (q-1)(q-5)(3q^2-28q+61) \\
C_5 &= 2^4 3^3 5^1 7^1 (q-1)(q-5)(q-7) \\
C_4 &= 2^2 3^2 5^1 7^1 (q-1)(q-5)(q-7)(4q-29) \\
C_3 &= 3^1 7^1 (q-1)(q-5)(q-7)(16q^2-77q-633) \\
C_2 &= 3^3 5^1 7^1 (q-1)(q-5)(q-7)(q-9)(q-11) \\
C_1 &= 3^2 7^1 (q-1)(q-5)(q-7)(q-9)(q-11)(q-13) \\
C_0 &= (q-1)(q-5)(q-7)(q-9)(q-11)(q-13)(q-17)
\end{aligned}
$$

$\bar{\chi}_{E_8}(q,t) = \sum_{i=0}^{120} C_i t^i$ where the C_i's are the following functions of q ($C_i = 0$ if not listed):

$$
\begin{aligned}
C_{120} &= 1 \\
C_{63} &= 2^3 3^1 5^1 (q-1) \\
C_{42} &= 2^3 3^3 5^1 (q-1) \\
C_{37} &= 2^5 3^1 5^1 7^1 (q-1) \\
C_{36} &= 2^5 5^1 7^1 (q-1)(q-5) \\
C_{30} &= 2^2 3^3 5^1 7^1 (q-1)(q-3) \\
C_{28} &= 2^6 3^3 5^1 (q-1) \\
C_{23} &= 2^5 3^3 5^1 7^1 (q-1) \\
C_{22} &= 2^8 3^3 5^1 (q-1) \\
C_{21} &= 2^4 3^3 5^1 37^1 (q-1)(q-5) \\
C_{20} &= 2^3 3^3 5^1 7^1 (q-1)(q-5)(q-7) \\
C_{16} &= 2^7 3^3 5^1 7^1 (q-1)(q-4) \\
C_{15} &= 2^5 3^2 5^1 7^1 (q-1)(q-5)(4q-23) \\
C_{14} &= 2^8 3^3 5^1 7^1 (q-1) \\
C_{13} &= 2^3 3^3 5^1 7^1 (q-1)(q-7)(5q+7) \\
C_{12} &= 2^1 3^2 5^1 7^1 (q-1)(q-7)(5q^2-80q+1091) \\
C_{11} &= 2^8 3^3 5^1 7^1 (q-1)(q-7)(q-8)
\end{aligned}
$$

$$
\begin{aligned}
C_{10} &= 2^7 3^3 7^1 (q-1)(q-7)(q^2-17q+97) \\
C_9 &= 2^6 3^3 5^2 7^1 (q-1)(q-7)(q-9) \\
C_8 &= 2^5 3^3 5^2 7^1 (q-1)(q-7)(3q-23) \\
C_7 &= 2^5 3^2 5^2 7^1 (q-1)(q-7)(q-11)(3q-19) \\
C_6 &= 2^3 3^1 5^1 7^1 (q-1)(q-7)(q-11)(9q^2-118q+893) \\
C_5 &= 2^6 3^3 5^2 7^1 (q-1)(q-7)(q-11)(q-13) \\
C_4 &= 2^3 3^2 5^1 7^1 (q-1)(q-7)(q-11)(q-13)(16q-179) \\
C_3 &= 2^3 5^1 7^1 (q-1)(q-7)(q-11)(q-13)(q-17)(4q+79) \\
C_2 &= 2^2 3^3 5^1 7^1 (q-1)(q-7)(q-11)(q-13)(q-17)(q-19) \\
C_1 &= 2^3 3^1 5^1 (q-1)(q-7)(q-11)(q-13)(q-17)(q-19)(q-23) \\
C_0 &= (q-1)(q-7)(q-11)(q-13)(q-17)(q-19)(q-23)(q-29)
\end{aligned}
$$

4.7.2 Tutte Polynomials

Using the transformation given earlier, the coboundary polynomials correspond to the following Tutte Polynomials:

$T_{E_6}(x,y) = y^{30}+6y^{29}+21y^{28}+56y^{27}+126y^{26}+252y^{25}+462y^{24}+792y^{23}+1287y^{22}+2002y^{21}+3003y^{20}+4368y^{19}+6188y^{18}+8568y^{17}+11628y^{16}+\sum_{i=0}^{15} D_i y^i$

where the D_i's are the following functions of x:

$$
\begin{aligned}
D_{15} &= 3(9x + 5159) \\
D_{14} &= 9(15x + 2243) \\
D_{13} &= 9(45x + 2863) \\
D_{12} &= 7(135x + 4591) \\
D_{11} &= 42(45x + 931) \\
D_{10} &= 6(573x + 7715) \\
D_{9} &= 10(585x + 5309) \\
D_{8} &= 45(x^2 + 208x + 1303) \\
D_{7} &= 90(2x^2 + 157x + 687) \\
D_{6} &= 18(37x^2 + 1113x + 3410) \\
D_{5} &= 18(98x^2 + 1457x + 3125) \\
D_{4} &= 45(83x^2 + 692x + 1025) \\
D_{3} &= 10(27x^3 + 657x^2 + 3222x + 3214) \\
D_{2} &= 30(27x^3 + 301x^2 + 922x + 570) \\
D_{1} &= 12(10x^4 + 155x^3 + 793x^2 + 1422x + 420) \\
D_{0} &= (x)(x + 3)(x + 4)(x + 6)(x + 7)(x + 10)
\end{aligned}
$$

$T_{E_7}(x,y) = y^{56}+7y^{55}+28y^{54}+84y^{53}+210y^{52}+462y^{51}+924y^{50}+1716y^{49}+3003y^{48}+5005y^{47}+8008y^{46}+12376y^{45}+18564y^{44}+27132y^{43}+38760y^{42}+54264y^{41}+74613y^{40}+100947y^{39}+134596y^{38}+177100y^{37}+230230y^{36}+296010y^{35}+376740y^{34}+475020y^{33}+593775y^{32}+736281y^{31}+\sum_{i=0}^{30} D_i y^i$ where the D_i's are the following functions of x:

$$
\begin{aligned}
D_{30} &= 28(x+32363) \\
D_{29} &= 84(2x+13183) \\
D_{28} &= 12(49x+112010) \\
D_{27} &= 56(28x+28943) \\
D_{26} &= 504(7x+3853) \\
D_{25} &= 504(14x+4587) \\
D_{24} &= 21(619x+130226) \\
D_{23} &= 126(179x+25509) \\
D_{22} &= 63(593x+59564) \\
D_{21} &= 4(14896x+1087739) \\
D_{20} &= 42(2191x+119246) \\
D_{19} &= 420(329x+13621) \\
D_{18} &= 14(14455x+462932) \\
D_{17} &= 504(575x+14439) \\
D_{16} &= 315(1291x+25690)
\end{aligned}
$$

$$
\begin{aligned}
D_{15} &= 6(63x^2 + 93358x + 1483763) \\
D_{14} &= 9(210x^2 + 84185x + 1075176) \\
D_{13} &= 630(9x^2 + 1597x + 16468) \\
D_{12} &= 14(945x^2 + 93650x + 782104) \\
D_{11} &= 84(315x^2 + 19955x + 135052) \\
D_{10} &= 84(583x^2 + 25007x + 136860) \\
D_9 &= 140(615x^2 + 18379x + 80996) \\
D_8 &= 15(21x^3 + 9660x^2 + 204366x + 721048) \\
D_7 &= 12(105x^3 + 19355x^2 + 294158x + 823882) \\
D_6 &= 42(123x^3 + 8450x^2 + 92669x + 203568) \\
D_5 &= 252(57x^3 + 2005x^2 + 16093x + 27240) \\
D_4 &= 315(99x^3 + 2116x^2 + 12456x + 15744) \\
D_3 &= 140(9x^4 + 441x^3 + 5724x^2 + 24356x + 21760) \\
D_2 &= 84(45x^4 + 1165x^3 + 9900x^2 + 29744x + 16896) \\
D_1 &= 24(14x^5 + 455x^4 + 5404x^3 + 28000x^2 + 55872x + 15360) \\
D_0 &= (x)(x+4)(x+6)(x+8)(x+10)(x+12)(x+16)
\end{aligned}
$$

$$
\begin{aligned}
T_{E_8}(x,y) = {} & y^{112}+8y^{111}+36y^{110}+120y^{109}+330y^{108}+792y^{107}+1716y^{106}+ \\
& 3432y^{105}+6435y^{104}+11440y^{103}+19448y^{102}+31824y^{101}+50388y^{100}+77520y^{99}+ \\
& 116280y^{98}+170544y^{97}+245157y^{96}+346104y^{95}+480700y^{94}+657800y^{93}+ \\
& 888030y^{92}+1184040y^{91}+1560780y^{90}+2035800y^{89}+2629575y^{88}+3365856y^{87}+ \\
& 4272048y^{86}+5379616y^{85}+6724520y^{84}+8347680y^{83}+10295472y^{82}+12620256y^{81}+
\end{aligned}
$$

$15380937y^{80}+18643560y^{79}+22481940y^{78}+26978328y^{77}+32224114y^{76}+38320568y^{75}+45379620y^{74}+53524680y^{73}+62891499y^{72}+73629072y^{71}+85900584y^{70}+99884400y^{69}+115775100y^{68}+133784560y^{67}+154143080y^{66}+177100560y^{65}+202927725y^{64}+231917400y^{63}+264385836y^{62}+300674088y^{61}+341149446y^{60}+386206920y^{59}+436270780y^{58}+491796152y^{57}+\sum_{i=0}^{56} D_i y^i$ where the D_i's are the following functions of x:

$$
\begin{aligned}
D_{56} &= 3(40x + 184423517) \\
D_{55} &= 24(35x + 25883968) \\
D_{54} &= 480(7x + 1450388) \\
D_{53} &= 1440(7x + 540816) \\
D_{52} &= 48(525x + 18116846) \\
D_{51} &= 528(105x + 1835888) \\
D_{50} &= 3168(35x + 340496) \\
D_{49} &= 22880(9x + 52376) \\
D_{48} &= 715(504x + 1858907) \\
D_{47} &= 264(2275x + 5574761) \\
D_{46} &= 12(80080x + 135584571) \\
D_{45} &= 408(3640x + 4401373) \\
D_{44} &= 90(24752x + 21986481) \\
D_{43} &= 2280(1428x + 954775)
\end{aligned}
$$

$$
\begin{aligned}
D_{42} &= 1140(4080x + 2097241) \\
D_{41} &= 6840(952x + 383251) \\
D_{40} &= 1045(8568x + 2745775) \\
D_{39} &= 8360(1449x + 375022) \\
D_{38} &= 25080(644x + 136345) \\
D_{37} &= 528(40250x + 7050797) \\
D_{36} &= 1716(16100x + 2357433) \\
D_{35} &= 216(164455x + 20311481) \\
D_{34} &= 1080(41867x + 4396801) \\
D_{33} &= 2160(26404x + 2374475) \\
D_{32} &= 27(2642360x + 204728791) \\
D_{31} &= 8(11072565x + 743009777) \\
D_{30} &= 4(280x^2 + 27310780x + 1594191303) \\
D_{29} &= 120(56x^2 + 1115912x + 56866291) \\
D_{28} &= 10(2352x^2 + 16324176x + 728329643) \\
D_{27} &= 8(7840x^2 + 24748885x + 968971896) \\
D_{26} &= 36(3920x^2 + 6638870x + 228489139) \\
D_{25} &= 24(11760x^2 + 11966640x + 362505737) \\
D_{24} &= 15(34748x^2 + 22907024x + 611314029) \\
D_{23} &= 360(2527x^2 + 1136905x + 26743289) \\
D_{22} &= 4(380205x^2 + 121329810x + 2516460976) \\
D_{21} &= 32(76685x^2 + 17898170x + 327354741)
\end{aligned}
$$

$$
\begin{aligned}
D_{20} &= 1176(3265x^2 + 571840x + 9222554) \\
D_{19} &= 1960(2982x^2 + 400665x + 5696603) \\
D_{18} &= 1960(4427x^2 + 465155x + 5827248) \\
D_{17} &= 10080(1249x^2 + 104350x + 1150796) \\
D_{16} &= 105(170364x^2 + 11480880x + 111298261) \\
D_{15} &= 120(63x^3 + 208244x^2 + 11432000x + 97204653) \\
D_{14} &= 180(210x^3 + 191167x^2 + 8606356x + 63992117) \\
D_{13} &= 280(405x^3 + 166977x^2 + 6193728x + 40116265) \\
D_{12} &= 14(18900x^3 + 4478240x^2 + 137288000x + 770870079) \\
D_{11} &= 168(3150x^3 + 493870x^2 + 12528895x + 60633356) \\
D_{10} &= 28(35460x^3 + 3869940x^2 + 81132870x + 336063047) \\
D_9 &= 280(6390x^3 + 497570x^2 + 8598812x + 30234267) \\
D_8 &= 45(70x^4 + 69440x^3 + 3900720x^2 + 55425456x + 163763985) \\
D_7 &= 72(175x^4 + 72450x^3 + 2992860x^2 + 34871505x + 85435721) \\
D_6 &= 252(221x^4 + 33476x^3 + 1017446x^2 + 9668906x + 19280614) \\
D_5 &= 504(317x^4 + 25402x^3 + 580362x^2 + 4468402x + 7060640) \\
D_4 &= 70(5031x^4 + 261216x^3 + 4522716x^2 + 27777024x + 33390904) \\
D_3 &= 280(27x^5 + 2763x^4 + 88308x^3 + 1131268x^2 + 5404792x + 4638096) \\
D_2 &= 168(135x^5 + 8095x^4 + 175220x^3 + 1642060x^2 + 5925840x + 3241632) \\
D_1 &= 16(70x^6 + 4935x^5 + 135282x^4 + 1801492x^3 + 11691432x^2 + 30438432x + 7983360) \\
D_0 &= (x)(x+6)(x+10)(x+12)(x+16)(x+18)(x+22)(x+28)
\end{aligned}
$$

4.8 Analyzing Results

4.8.1 Patterns

Upon looking both the coboundary polynomials and the Tutte polynomials, we notice some interesting patterns. First, all coefficients in the coboundary polynomial are divisible by $q-1$ except for $C_d = 1$, where d is the number of hyperplanes of the arrangements. We can see why this would be true from the definition:

$$\bar{\chi}_A(q,t) = \sum_{\substack{B \subseteq A \\ \text{B central}}} q^{r(A)-r(B)}(t-1)^{|B|}$$

so when $q = 1$,

$$\begin{aligned} \bar{\chi}_A(1,t) &= \sum_{\substack{B \subseteq A \\ \text{B central}}} (t-1)^{|B|} \\ &= \sum_{k=0}^{|A|} \binom{|A|}{k} (t-1)^k \\ &= (t-1+1)^{|A|} = t^{|A|} \end{aligned}$$

where the second step follows because there are $\binom{|A|}{k}$ subsets B of size k, and the third step follows from the binomial theorem. Thus we see that $q-1$ will divide all but the highest order term.

We also notice that the coboundary polynomials appear to have a much smaller height than the Tutte polynomials. This would be a something to

examine in future research.

4.8.2 Polynomials from Invalid Primes

It is interesting to consider the results that we do get from primes where there is an invalid reduction. One way to confirm that the reduction is invalid is to consider the correct coboundary polynomial which was found using valid reductions. We note that this correct polynomial evaluated at invalid primes often has negative terms in them, which is not achievable using our counting method, so the method clearly cannot be correct there.

In these situations, we are usually getting degenerate hyperplanes and hyperplanes are overlapping, and so the result is that the polynomials found using our method are missing the lower order terms. The higher order terms are usually correct. As an example, when run on E_7 on $q = 3$, our algorithm found the polynomial $f(t) = t^{63} + 56t^{36} + 126t^{30} + 1332t^{21} + 672t^{18}$. However, here is the actual coboundary polynomial:

$$\begin{aligned}\bar{\chi}_{E_7}(3,t) &= t^{63} + 56t^{36} + 126t^{30} + 1332t^{21} + 2016t^{16} - 1344t^{15} + 4032t^{13} \\ &- 12096t^{11} + 18144t^{10} - 20160t^9 - 30240t^8 + 40320t^7 - 6720t^6 \\ &+ 241920t^5 - 342720t^4 - 241920t^3 + 725760t^2 - 483840t + 107520\end{aligned}$$

4.8.3 Validity of Results

We can use known specializations of the Tutte Polynomial to get confirmation that this is correct. For example, we know that when $y = 0$ in $\bar{\chi}(q,t)$, we get

the chromatic polynomial. The values for C_0 in each case above correspond to these known values (see [5], p. 59).

We also compared these polynomials with those found by de Concini and Procesi using the method described earlier, and the polynomials were identical ([3]).

4.9 Further Applications

We used this algorithm to compute the coboundary polynomial of three specific arrangements, but it could be used similarly on other arrangements. In this particular case, we were able to compute these coboundary polynomials using only primes up to 13. Without this method, we would have needed to use eight valid primes to compute E_8, which means we would have needed to use primes $7 \leq p \leq 31$, and thus our method gave a substantial computational savings.

One important feature of these particular arrangements was the fact that they had very few and very predictable subarrangements. This method could also be potentially used for other computations of similar arrangements where characteristics of a large arrangement can be determined by characteristics of its subarrangements.

Appendix

Appendix A

GP Code

(Note: An electronic version of each of these functions can be found online.)

A.1 Initial Data

This function is used to create the matrix to represent the E_8 arrangements. It first defines the planes in $\mathbb{R}^8$ and then writes them in a basis of the simple roots described earlier. The functions for E_7 and E_6 are similar and thus not shown, but can be found in createEPlanes.gp.

```
createE8()=
{
        \\We will write E8 in terms of this basis
    ChangeOfBasis=[1,-1,-1,-1,-1,-1,-1,1;
        2,2,0,0,0,0,0,0;
        -2,2,0,0,0,0,0,0;
        0,-2,2,0,0,0,0,0;
        0,0,-2,2,0,0,0,0;
        0,0,0,-2,2,0,0,0;
```

```
    0,0,0,0,-2,2,0,0;
    0,0,0,0,0,-2,2,0]~;

OrigPts = vector(120,loop,[0,0,0,0,0,0,0,0]);
row=1;

for(i=1,8,
    for(j=i+1,8,
        OrigPts[row][i]=1;
        OrigPts[row][j]=1;
        row++));

for(i=1,8,
    for(j=i+1,8,
        OrigPts[row][i]=-1;
        OrigPts[row][j]=1;
        row++));

allsigns = vector(7,loop,[0,1]);
forvec(signs=allsigns,
    if(sum(i=1,7,signs[i])%2==0,
        for(loop=1,7,OrigPts[row][loop]=(-1)^signs[loop]);
        OrigPts[row][8]=1;
        row++));
```

```
    E8Planes=vector(120,i,[]);
    for(row=1,120,
        newpts=matsolve(ChangeOfBasis,OrigPts[row]~);

    putInList=vector(8);
    half=0;
    for(i=1,8,putInList[i]=newpts[i];
        if((newpts[i]*2)%2==1,half=1));

    if(half==1,putInList=putInList*2);

    E8Planes[row]=putInList);

    E8Planes
}
```

The following two functions are used to create the matrices to represent the A and D arrangements.

```
\\Create the matrix for the A_n arrangement
createA(n)=
{
    APlanes = vector(binomial(n+1,2));
```

```
    rownum=1;
    for(i=1,n+1,
        for(j=i+1,n+1,
            row = vector(n+1,x,0);
            row[i]=1;
            row[j]=-1;
            APlanes[rownum] = row;
            rownum++));

    APlanes
}

\\Create the matrix for the D_n arrangement
createD(n)=
{
    DPlanes = vector(2*binomial(n+1,2));

    rownum=1;
    for(i=1,n+1,
        for(j=i+1,n+1,
            row = vector(n+1,x,0);
            row[i]=1;

            row[j]=-1;
```

```
            DPlanes[rownum] = row;
            rownum++;

            row[j]=1;
            DPlanes[rownum] = row;
            rownum++));

    DPlanes
}
```

These functions use Ardila's formula to find the coboundary polynomials and associated data for all A_n and D_n arrangements that will show up as subarrangements later on. It will write them to the file "ABdata" for easy reference.

```
GetAn(n)=
{
    MS=serlaplace(sum(n=0,20,t^binomial(n,2)*x^n/n!,O(x^21))^q);

    poly=polcoeff(MS,n+1,x)/q;

    write(ADdata,"\\\\poly, dim, number of dependencies");
    write(ADdata,"A[",n,",1]= ",poly,";");
    write(ADdata,"A[",n,",2]= ",binomial(n+1,2),";");
    write(ADdata,"A[",n,",3]= ",binomial(n+1,3),";");
```

```
    write(ADdata,"");

    poly
}

GetDn(n)=
{
    sum1=sum(n=0,20,2^n*t^binomial(n,2)*x^n/n!,O(x^21))^((q-1)/2);
    sum2=sum(n=0,20,t^(n*(n-1))*x^n/n!,O(x^21));
    MS=serlaplace(sum1*sum2);

    poly=polcoeff(MS,n,x);

    write(ADdata,"\\\\poly, dim, number of dependencies");
    write(ADdata,"D[",n,",1]= ",poly,";");
    write(ADdata,"D[",n,",2]= ",2*binomial(n,2),";");
    write(ADdata,"D[",n,",3]= ",4*binomial(n,3),";");
    write(ADdata,"");

    poly
}
```

A.2 Calculation for Primes

This function uses Ardila's Theorem to compute the Tutte Polynomial for an arrangement (here, E_8) over some $\mathbb{F}_p$. Again, this function is similar for all arrangements, and they can all be found in Primes.gp.

```
E8(p)=
{
    local(allx,x,loop,cnt,poly);

    poly=0;
    PLANES = E8Planes;

  \\Loop through all points, counting the
  \\hyperplanes containing each point
    allx = vector(8,loop,[1,p]);
    forvec(x=allx,
        cnt=0;
        for(loop=1,length(PLANES),
            if((PLANES[loop]*mattranspose(x))%p==0, cnt +=1));
        poly += t^cnt);

    poly
}
```

A.3 Subarrangements

The following function finds the coboundary polynomials for all possible subarrangements of E_6, E_7, and E_8. If we are looking at the arrangement E_n, the outer loop represents the dimension of the subarrangements, which is between 1 and $n-1$. For each dimension d, the next loop looks at all possible partitions of d. For each partition $d = d_1 d_2 \dots d_k$, where $d_i \geq d_{i+1}$, there are are two possible types of collections of subarrangements: either the direct sum $A_{d_1} \oplus A_{d_2} \oplus \cdots \oplus A_{d_k}$, or $D_{d_1} \oplus A_{d_2} \oplus \cdots \oplus A_{d_k}$. The final subarrangements that must be added are those that contain some E_n. There are only a few of these, so they are simply added separately. As previously mentioned, note that we must include the entire arrangement as a subarrangement.

For each subarrangement, we do three calculations. The first is the coboundary polynomial, which is simply the product of the coboundary polynomial of all elements of the direct sum. Our goal is to easily determine which subarrangement we have at any given time, so we calculate the number of planes of the subarrangement as well as the number of sets of three dependent planes. We then store this in a matrix, indexed by the number of planes of the subarrangement, for efficiency.

```
CreateSubPolys(p,dim,numPlanes)=
{
\\Uses dim to calculate "degrees of freedom", i.e. the mult factor
\\And to see how far to go
```

```
local(A,D,q,curr,subdim,allparts,subarr,element,subPoly,loop,loop2);

subPoly=matrix(120,2,loop,loop2,[]);

A=getAmatrix();  D=getDmatrix();
for(subdim=1,dim-1,  \\Do it for each dimension

    allparts=part(subdim);
    for(subarr=1,length(allparts),  \\For each partition,

        \\Do the A's
    curr=[1,0,0];
    for(element=1,length(allparts[subarr]),
        curr[1]*= getApoly(p,allparts[subarr][element]);
        curr[2]+= A[allparts[subarr][element],2];
        curr[3]+= A[allparts[subarr][element],3]);

    subPoly[curr[2],1]=concat(subPoly[curr[2],1], curr[1] * p^(dim-subdim));
    subPoly[curr[2],2]=concat(subPoly[curr[2],2], curr[3]);
        \\Check for D's
    if(allparts[subarr][1]>=4,
        curr[1]= getDpoly(p,allparts[subarr][1]);
        curr[2]= D[allparts[subarr][1],2];
        curr[3]= D[allparts[subarr][1],3];
```

```
            for(element=2,length(allparts[subarr]),
                curr[1]*= getApoly(p,allparts[subarr][element]);
                curr[2]+= A[allparts[subarr][element],2];
                curr[3]+= A[allparts[subarr][element],3]);

        subPoly[curr[2],1]=concat(subPoly[curr[2],1], curr[1] * p^(dim-subdim));
        subPoly[curr[2],2]=concat(subPoly[curr[2],2], curr[3])));

\\These are the E's, including the one we're calculating
    if(dim==6,
        subPoly[36,1]=concat(subPoly[36,1],[f[p]] * p^(dim-6)),
    if(dim==7,
        subPoly[36,1]=concat(subPoly[36,1],[E6(t,p)] * p^(dim-6));
        subPoly[63,1]=[g[p] * p^(dim-7)],  \\get the precomputed E7 for p
    if(dim==8,
        subPoly[36,1]=concat(subPoly[36,1],[E6(t,p)] * p^(dim-6));
        subPoly[37,1]=concat(subPoly[37,1],[E6(t,p)*A1(t,p)] * p^(dim-7));
        subPoly[63,1]=concat(subPoly[63,1],[E7(t,p)] * p^(dim-7));
\\get the precomputed E8 for p
        subPoly[120,1]=concat(subPoly[120,1],[h[p]] * p^(dim-8)))));

\\Used to return a matrix of the correct size
        returnPoly=matrix(numPlanes,2,loop,loop2,[]);
```

```
        for(i=1,numPlanes,for(j=1,2,returnPoly[i,j]=subPoly[i,j]));

    returnPoly
}
```

This is the main function that calculates the coboundary polynomial over $(\mathbb{Z}/pq\mathbb{Z})$. It is shown here for E_8, but see GetCoPoly.gp for all arrangements. It loops through all points mod p_1, and for each point uses the subarrangement calculations to determine what the coboundary polynomial is for the cosets of that point. There are several helper functions shown below that aid in distinguishing between subarrangements.

```
\\For E8
GetCoPoly(p1,p2,n)=
{
    local(allx0,allx1,x0,loop,loop2,cnt,poly,dim);

    getdata();      \\from ADdata

    poly=0;

    dim=8;
    numPlanes=120;
    PLANES=E8Planes;
```

```
subPoly = CreateSubPolys(p2,dim,numPlanes);

    \\Find the planes for the original point
allx0 = vector(dim,loop,[1,p1]);
forvec(x0=allx0,            \\x0 is just mod p1

    candidatePlanes=[];
    for(loop=1,numPlanes,
        if((PLANES[loop]*mattranspose(x0))%p1==0,
            candidatePlanes=concat(candidatePlanes,[PLANES[loop]])));
    numCandidatePlanes = length(candidatePlanes);

    \\Find the poly for the translates of the point
if(numCandidatePlanes==0, poly += p2^dim,

    if(length(subPoly[numCandidatePlanes,1])<>1,
     numdep=samesummod2(candidatePlanes,numCandidatePlanes,dim);
     index = getindex(subPoly,numCandidatePlanes,numdep);
     currPoly = subPoly[numCandidatePlanes,1][index],
     currPoly = subPoly[numCandidatePlanes,1][1]); \\if len=1

    poly += currPoly));

poly
```

```
}


\\This determines the number of sets of three dependent planes
samesummod2(mat,size,dim)=
{
    count=0;
    for(row1=1,size,
        for(row2=row1+1,size,
for(row3=row2+1,size,
            if((mat[row1]+mat[row2]+mat[row3])%2==0,count+=1))));

    count
}


\\This determines what entry of the subpoly matrix to use
getindex(subPoly,row,numdep)=
{
    index=0;

    for(indexloop=1,length(subPoly[row,2]),
        if (subPoly[row,2][indexloop] == numdep, index=indexloop));
```

```
    index
}
```

Bibliography

[1] Federico Ardila. Computing the Tutte polynomial of a hyperplane arrangement. *Pacific Journal of Mathematics*, 230:1–26, 2007.

[2] Christos A. Athanasiadis. Characteristic polynomials of subspace arrangements and finite fields. *Advances in Mathematics*, 122:193–233, 1996.

[3] C. De Concini and C. Procesi. The zonotope of a root system. *Transformation Groups*, 13:507–526, 2008.

[4] Michael Eastwood and Stephen Huggett. Euler characteristics and chromatic polynomials. *European Journal of Combinatorics*, 28:1553–1560, 2007.

[5] James E. Humphreys. *Reflection Groups and Coxeter Groups*. Cambridge University Press, 1990.

[6] Nicholas M. Katz. Appendix: E-polynomials, zeta-equivalence, and polynomial-count varieties.

[7] Peter Orlik and Hiroaki Terao. *Arrangements of Hyperplanes*. Springer-Verlag, 1992.

[8] Victor Reiner. An interpretation for the Tutte polynomial. *European Journal of Combinatorics*, 20:149–161, 1999.

[9] W. Kook; V. Reiner; and D. Stanton. A convolution formula for the Tutte polynomial. *Journal of Combinatorial Theory Series B*, 76:297–300, 1999.

[10] W. T. Tutte. A contribution to the theory of chromatic polynomials. *Canadian Journal of Mathematics*, 6:80–91, 1954.

[11] W. T. Tutte. *Introduction to the Theory of Matroids*. American Elsevier Publishing Company, Inc., 1971.

[12] D.J.A. Welsh and C. Merino. The Potts model and the Tutte polynomial. *Journal of Mathematical Physics*, 41:1127–1152, 2000.

[13] Wikipedia. Partition of a set, 2009. [Online; accessed 1-May-2009].

[14] Wikipedia. Root system, 2009. [Online; accessed 1-May-2009].

Vita

Todd Geldon was born in Washington, D.C. in 1977. After completing his high school studies in Silver Spring, Maryland, in 1995, he entered Princeton University. He received a Bachelor of Arts (A. B.) in Mathematics from Princeton University in 1999. In August 2001, he entered the Graduate School of the University of Texas at Austin.

Permanent address: 4516 Avenue D Apt B
Austin, Texas 78751

This dissertation was typeset with LaTeX[‡] by the author.

[‡]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.